Woods Hole Oceanographic Institution

WHOI-95-06

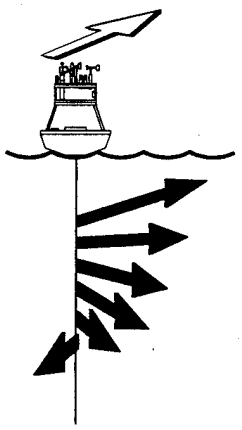
Technical Report

March 1995

A Processing System for Argos Meteorological Data

by

Nancy R. Galbraith



Upper Ocean Processes Group
Woods Hole Oceanographic Institution
Woods Hole, Massachusetts 02543

UOP Technical Report 95-3

WHOI-95-06
UOP-95-03

A Processing System for Argos Meteorological Data

by

Nancy R. Galbraith

Woods Hole Oceanographic Institution
Woods Hole, Massachusetts 02543

March 1995

Technical Report

Funding was provided by the Office of Naval Research through Grant No. N00014-94-1-0161.

Approved for Distribution:

Philip L. Richardson, Chair
Department of Physical Oceanography

A Processing System for Argos Meteorological Data

Nancy R. Galbraith
Upper Ocean Processes Group
Physical Oceanography Department
Woods Hole Oceanographic Institution

13 March 1995

Abstract

Upper Ocean Processes Group meteorological data is transmitted from surface buoys via Argos satellite and processed in an automatic mode on a UNIX workstation. Data is extracted from input files based on instrument type and experiment, processed as appropriate, and plotted, without user intervention. While the processing system normally runs automatically, it is designed so that modules can also be run directly from a terminal when necessary. The Argos processing system allows us to monitor the meteorological data being collected in the field, and provides early information about problems with sensors, instruments, or buoys, when they occur. The automatic process allows more information to be viewed with less effort, and increases the usefulness of the Argos data.

Contents

Abstract i

1 UNIX Implementation and Overview 1
1.1 Cron 2

2 Main Processing Script 3

3 Processing VAWR Data 4
3.1 Shell Script gargle.sh 5
3.2 Program Gargle 5
3.3 VAWR Table Files 6

4 Processing IMET Data 7
4.1 Shell Script Imet.sh 8
4.2 Program Gargle_imet 9
4.3 IMET Table Files 9

5 Processing Engineering Variables 10

6 Processing Position Data 11

7 Data Transmission Scripts 12

8 Plotting Scripts 13
8.1 Plot Plus Time Series Plots 13
8.2 Matlab Position Plot 14

9 Appendices 15
Appendix I. Startup Shell Scripts 15
Appendix II. VAWR Processing 19
Appendix III. IMET Processing 25
Appendix IV. Engineering PTT Processing 33
Appendix V. Position Processing Scripts 38
Appendix VI. Data Distribution Scripts 41
Appendix VII. Plotting Scripts and Sample Plots 43

Acknowledgements **54**

1 UNIX Implementation and Overview

Data transmitted from surface buoys via Argos satellite for the Upper Ocean Processes (UOP) Group experiments are processed in an automatic mode on a Sun workstation, Griffon, using a series of shell scripts and C programs initiated by the UNIX cron utility. Files of data from all of the UOP's active Argos transmitters are sent daily from Service Argos' computers in Landover, Maryland, using ftp. The UNIX cron utility runs a script every hour which checks for incoming data. When a new data file is found, processing scripts are activated.

Each platform transmitter terminal, or PTT, in the field includes its own identification number in the messages it sends to Service Argos. This number allows Service Argos to route each message to the correct destination, grouping the messages it receives for retransmission to the PTT owner. Each PTT identification number is associated with an experiment number, which is used for routing and accounting purposes, and so data for each experiment is bundled into a separate file. Currently, we receive data from our Arabian Sea buoy in a file called argos.dat, and data from our GLOBEC buoy in file argos2.dat.

Incoming data is processed by a suite of C programs and shell scripts. While these programs and scripts are normally run automatically by the UNIX scheduler, cron, they are designed so they can also be run directly from a terminal in cases where Argos transmissions are interrupted, an instrument must be reprocessed, or non-standard plots are desired.

The primary goal of the Argos processing system is to monitor the meteorological data being collected in the field, as soon as it becomes available. The system also provides early information about problems with sensors, instruments, or buoys, when they occur. By automating the entire process, we allow more information to be viewed with less effort, and increase the usefulness of the Argos data.

1.1 Cron

The UNIX cron utility is currently run by user nan on Griffon. Cron executes commands at specific times. The commands are entered into a table, called the cron table, by program crontab. The cron table entry on Griffon can be edited by its owner by typing crontab -e. New cron table entries can be started using crontab -e, or checked by typing crontab -l. The cron table entry for the Arabian Sea and GLOBEC Argos processing is:

```
1 * * * * /gdata/argos/argos.sh
1 8 * * 1-5 /puli/data/arab/argos/dopl.sh
1 8 * * 1-5 /puli/data/globec/argos/dopl.sh
```

The first entry runs the script argos.sh at one minute past every hour, every day. The second and third commands generate plots at 8 a.m. on weekdays only, using plotting scripts for two experiments in their processing directories.

The crontab entries are basically names of shell script programs. To change the processing system, the scripts themselves can be edited, without having to alter the crontab entry. As experiments start and end, the appropriate commands can be added to or deleted from the main processing scripts.

The directory /gdata/argos on Griffon contains all the shared processing software and has subdirectories for each active experiment. These experiment subdirectories contain archives of Argos data and information about the instrumentation. The processed data normally resides, along with experiment-specific scripts, on another workstation, on a disk that is mounted on Griffon using the Network File System (NFS).

If data must be retransmitted from Service Argos for any reason, argos.sh will automatically process any new data every hour. As long as Service Argos does not send more than one file within that time period, cron will cause argos.sh to copy and process all incoming data files with the names

specified in the script. To process an incoming file immediately, the script argos.sh can be invoked from the keyboard.

2 Main Processing Script

The main processing is initiated by the shell script argos.sh, in Griffon's directory /gdata/argos. Incoming Argos data is written to that directory in files named argos.dat and argos2.dat, which represent two different Argos accounts. This script handles each input file separately, first processing all data in argos.dat.n, then argos2.dat.n.

For each experiment, the script first checks for the existence of the appropriate incoming data file. If the file exists, it is moved to a file named argos(2).dat.n, where n is a monotonically increasing number. Note that if the file argos.dat.1 does not exist, that filename is used for the new file. If it does exist, the directory is checked by the shell script findlast.sh to determine the next sequence number for the output filename, which is numerically one greater than the highest existing number. Therefore, removing the file argos.dat.1 will cause erratic numbering, since the next file processed will be called argos.dat.1, but incoming data files will never be overwritten by the automated processing system.

Once the file archiving is done, the script processes data from the Vector-Averaging Wind Recorder (VAWR), then data from the Improved Meteorological (IMET) System, if it is being transmitted. If IMET data is not transmitted, there is normally a tensiometer reporting, and its data is processed after the VAWR data. Position data, generated by the Argos system, is processed last. Processing each of these data sets requires a separate pass through the incoming file, looking for the appropriate Argos transmitter ids. Each of these tasks is handled by an independent shell script, which can be run manually if needed.

See Appendix I for text of the shell scripts argos.sh and findlast.sh.

3 Processing VAWR Data

The script argos.sh processes VAWR data using another shell script, gargle.sh, which in turn runs a C program to extract and scale the incoming Argos data. Shell script /gdata/argos/gargle.sh starts one of the gargle processing programs, which extract VAWR data from a specified Argos input file based on information in a calibration file.

The VAWR data files produced by this system contain the variables yearday, wind east, wind north, wind speed, wind direction, short wave radiation, relative humidity, barometric pressure, sea temperature, air temperature, long wave thermopile voltage, body temperature, dome temperature, and long wave radiation. Air and sea temperatures are recorded in degrees centigrade, while body and dome are Kelvin temperatures. Wind vectors are recorded in meters per second, radiation values in watts per square meter, and thermopile voltage in microvolts. Relative humidity is reported as a percentage, and barometric pressure is reported in millibars. A sample output file is contained in Appendix II.

This system uniformly uses the convention that yearday 1 begins at midnight on January 1. This is consistent with the UNIX date utility, which is used to check for invalid dates and to set date limits for plots.

Occasionally VAWR data will need to be reprocessed. This will usually occur when corrections need to be made to a table file. If a VAWR's interval counter resets, the date calculated by the software system will be incorrect, and the data will need to be reprocessed after a table file change has been made. To reprocess VAWR data in case of errors or resets, remove any unwanted or incorrect data from the file /gdata/argos/exper/vawrnnnn.arch, which is appended by the processing system. Then cd to /gdata/argos and type

```
gargle.sh garglenew argos.dat.nn exper vawrnnnn >>
                        /gdata/argos/vawrnnnn.log 2>&1
```

where nn is the number of the argos.dat file to be reprocessed, exper is the experiment name and nnnn is the 4-digit VAWR number. The experiment name must correspond to two data directories, one in /puli/data and one in /gdata/argos. The first, /puli/data/exper, must have a subdirectory named argos, and the second, /gdata/argos/exper, must contain a table file for the VAWR to be processed.

Note that the working data file in the processing subdirectory on Puli is overwritten daily by the automated processing system, so any changes made there will be lost. To permanently alter the working data file, the archive version of the file, in the /gdata/argos directory, must be modified.

3.1 Shell Script gargle.sh

Arguments to gargle.sh are the gargle program name, the input data file to be processed, the experiment, and the VAWR name. Using these arguments, gargle.sh generates filenames and commands to process the incoming data.

Gargle.sh calls program gargle or garglenew with the filenames of the input Argos file, the VAWR table file and the output VAWR data file. Then gargle.sh sorts the new data and appends it to an archive file in the experiment subdirectory on Griffon. The archive file is then sorted by date with the UNIX sort utility, deleting duplicate entries. The sorted version of the file is placed in the processing directory for the experiment on Puli, in a file names vawrnnnn.asc.

See Appendix II for text of gargle.sh.

3.2 Program Gargle

Program gargle extracts VAWR data from a specified Argos input file based on information in a calibration file, also called a table file. The table file

contains the key to variable positions within the incoming records, allowing data to be extracted, and contains calibration values to be used in scaling the data. Program gargle writes data to a new file, and will overwrite an existing file if one is specified as the output file.

Gargle was adapted by Roger Goldsmith, of WHOI's Computer and Information Services (CIS) group.The original version, called gargoyle, was written by Thomas Danforth to run under the SCO Zenix operating system on a 386 PC, and was designed to work with a commercial database. Program gargle, which runs under SunOS UNIX, uses portable ASCII data files. The specifics of program gargle are beyond the scope of this manual.

There are at present two versions of the gargle program. Programs gargle and garglenew are identical except for the handling of long wave radiation. Both versions calculate long wave from independently scaled thermopile voltage and body and dome temperatures, but garglenew uses a new algorithm. Because the new type of calibration constants were not available for the Arabian Sea VAWR, we retained the original code for the duration of that experiment. The old version will be removed after the first Arabian Sea mooring is recovered.

3.3 VAWR Table Files

Program gargle uses table files which have the same format as those used by the tape processing programs vawr_cdf and vawr_cal, but with several extensions needed to decode the Argos record. The table files control the processing of the incoming record and provide documentation of calibrations used. The use of table files provides some flexibility to the system, accommodating changes in the incoming record.

Gargle extracts the PTT numbers from the specified table file, and uses those numbers to decide which records to extract from the incoming Argos file. It also uses the position of a variable description in the table file and the length in bits specified to determine the position and length of the variable within the Argos record. The calibration values are used by gargle

to scale the raw data values, using the same algorithms as those used by the tape processing system. The minimum and maximum values for each variable specified in the table file prevent wild values from entering the data stream. This windowing can mask problems with sensors, and must be used very carefully, especially during instrument evaluation.

Comment lines in table files are denoted with a two-part forward arrow, consisting of a dash and a greater-than symbol. Variable identification lines begin with a number sign, an integer representing the variable position, a colon, and the short and long version of the name of the variable. A variable's short name is used to select the processing function to be used on the data in the field defined by the entry, and should not be modified.

See Appendix II for a sample VAWR table file.

4 Processing IMET Data

The IMET system in the GLOBEC experiment transmits hourly averages to Argos. Script argos.sh processes the IMET data, as it does for the VAWR data, by invoking a secondary shell script which in turn calls a C program. The shell script imet.sh calls program gargle_imet, which extracts data from the incoming file using information from a table file.

The IMET system software has the capability to handle many variables, some of which have not been implemented in the IMET hardware at this time. The program which decodes the input file produces an intermediate file, which is modified by the awk script imet.awk before being archived and used. In the awk script, the variable order is changed to more closely resemble the order of variables in VAWR files. The final output file has a format which is similar to the VAWR files, but has several dummy variables, which are set to 0.

The variables in the working or archive version of the IMET files are real yearday, wind east, wind north, dummy, dummy, short wave radiation,

relative humidity, barometric pressure, air temperature, sea temperature, dummy, dummy, dummy, long wave radiation, battery voltage, mooring tension, and precipitation. The units for variables which are reported by both VAWR and IMET are described above. In addition, the IMET battery voltage is recorded in amps, the mooring tension in pounds, and the precipitation sensor measures cumulative precipitation in millimeters. A sample output file is found in Appendix III.

To reprocess IMET data in case of errors, use a text editor to remove unwanted data from file imet1.arch in the experiment subdirectory in /gdata/argos. This file is appended by script imet.sh. After any unwanted or erroneous data has been removed, cd to /gdata/argos and type

```
imet.sh argos2.dat.nn exper instrument
```

where nn is the number of the argos2.dat file to be reprocessed, exper is the experiment name, and instrument is the instrument name, normally imet1 or imet2. The experiment name must correspond to two data directories, one in /puli/data and one in /gdata/argos. The first, /puli/data/exper, must have a subdirectory named argos, and the second, /gdata/argos/exper, must contain a table file with the name of the instrument and the filename extension .tbl. Note that changes to the working data file in the experiment directory on Puli will be overwritten by the automated processing.

4.1 Shell Script Imet.sh

Arguments to imet.sh are the input filename, the experiment/directory, and the instrument name. Shell script imet.sh invokes program gargle_imet with the name of the Argos input file, the table file and the output file. The script runs the gargle output through an awk script to compute real yearday, reorder the variables and remove data with invalid dates. The awk script also adjusts the time of the data records to the half hour by adding

30 minutes to the reported time, and normalizes air temperatures if needed. The shell script then sorts the modified data, appends it to the archive file, sorts the archive file and copies it to the experiment's processing subdirectory in /puli/data.

See Appendix III for text of imet.sh and imet.awk.

4.2 Program Gargle_imet

Gargle_imet extracts the PTT numbers from the specified table file, and retrieves records with matching PTT numbers from the Argos file. The input records are decoded based on the length and position of each variable specified in the IMET table file. Data is converted from ASCII hexadecimal to floating point and written out in comma-separated ASCII strings. Output is to a new file, and, if an existing file is specified, it will be overwritten.

The variables output by program gargle_imet in the present implementation are: Integer day, hour, barometric pressure, air temperature, sea temperature, wind east, wind north, relative humidity, short wave radiation, long wave radiation, precipitation, battery voltage, and mooring tension. The order of these variables is modified by the shell script imet.sh, as described above.

4.3 IMET Table Files

The IMET table files are similar to the VAWR table files used by program gargle. Because IMET sensors return calibrated data in scientific units, the IMET table files do not contain calibration coefficients for most variables. They do contain information needed to decode and scale the incoming data and to control the processing itself.

At present, each Argos IMET record contains two IMET data records. To

accommodate that double record in one pass by program gargle_imet, the table file essentially contains two complete, identical descriptions of the IMET record. See Appendix III for a sample of an IMET table file.

5 Processing Engineering Variables

For some experiments, a separate Argos transmitter reports mooring tension from an independent tensiometer. Other variables, of interest for engineering purposes, may be included in the Argos record for these transmitters. For the Arabian Sea I experiment, variables transmitted are tension and battery voltage.

Shell script dotens.sh in directory /puli/data/arab/argos processes the data from the engineering Argos transmitter. This script is normally called with the number of the argos.dat.n file to be processed. If it is called with no arguments, it will process the most recent argos.dat.n file in directory /gdata/argos.

Dotens.sh uses the awk script tens.awk to identify tensiometer records in the incoming Argos file and to calculate year day. These fields are written to file tens.raw. Tension and battery voltage are ASCII hexadecimal fields, and are converted to decimal values and scaled in program conv. Conv writes the working tension file, tens.asc, which contains real yearday, year, month, day, hour, minute, second, tension, dummy, and battery voltage.

To reprocess tension data in case of errors, go to the processing directory for the experiment, usually on Puli. If necessary, remove any incorrect data from the database by editing file tens.raw, which is appended by this script, then type

```
dotens.sh nn
```

where nn is the number of the argos.dat file you wish to reprocess.

See Appendix IV for the text of dotens.sh, tens.awk, and conv.c.

6 Processing Position Data

Position data is processed by a separate script, and stored in a subdirectory under the usual processing directory for Argos data for each experiment. The script that processes Arabian Sea I position data, for example, is doposit.sh, and is found in /puli/data/arab in the argos/posit subdirectory. Position data is generated by Service Argos and incorporated into the incoming VAWR and IMET records.

Unlike meteorological data, position data seldom needs to be reprocessed, as it is not liable to contain timing errors and is not subject to calibration. However, it is often desirable to remove portions of the position record for an experiment, such as the steaming time before a mooring deployment. This allows plotting with auto-ranging axis limits, at an expanded scale, so that movement of a mooring can be monitored easily.

Argos position records contain a value indicating the quality of the calculated position, a number between one and three, with three as the highest quality. In this processing scheme, all location records are used, but the quality word is retained for future use. Including all position values, the Argos positions seem to be accurate to approximately one kilometer.

To reprocess position data, use a text editor to remove unwanted values, if necessary, from archive files in /puli/data/exper/argos. For GLOBEC, which generates position records for both IMET and VAWR PTTs, these files are in /puli/data/globec/argos/posit and are named v1pos.arch.asc and i1pos.arch.asc. For the Arabian Sea deployment, which receives only VAWR positions, the position file is pripos.arch.asc. After removing any incorrect data, cd to /puli/data/exper/argos/posit and reprocess by typing

```
doposit.sh nn
```

where nn is the number of the argos(2).dat file you wish to rerun. If this argument is omitted, doposit will process the most recent Argos file for the experiment related to the current working directory.

Position data is extracted from the Argos file by finding records in the file with a line length of 13 words and piping those records to an awk script, posit.awk. That script calculates real yearday, normalizes longitude to 180 degrees, and writes out PTT number, year, month, day, hour, minute, second, yearday, lat, long and a quality flag. This data is written to the temporary file junk.dat. The shell script then checks this file for the desired PTT numbers, and if any exist, appends the position record, stripped of the PTT number, to the appropriate archive file. This file is then sorted and the output written over the working version of the position file. For the Arabian Sea experiment, that file is /puli/data/arab/argos/posit/arab1.pos.

See Appendix V for the text of doposit.sh and posit.awk.

7 Data Transmission Scripts

To automatically distribute data to hosts with anonymous ftp accounts, ftp is run from the shell script ftp.sh, invoked by argos.sh. The script uses the -n and -i flags, which allow non-interactive use of ftp, redirecting input commands from the text of the script itself. For sites receiving data via ftp, we transmit the entire working version of the data every day, overwriting the data previously sent.

For hosts without anonymous ftp sites, we use the UNIX mail utility to distribute the data. When using mail, we send only updates to the data. Since we want to be sure to send a full day's worth of data, we use the UNIX utility tail to extract the last 100 records from the working version of the data files. In the current implementation, we extract variables of interest for our mail recipients, using an awk script, x.awk. We prepend a header record identifying the data as ours, and a record containing the variable names, before invoking mail.

See Appendix VI for the text of the scripts used to distribute Argos data. Note that the hosts names and IP addresses have been changed in the scripts reproduced there, for security purposes. These scripts can be run manually to retransmit data if necessary.

8 Plotting Scripts

Plots of all incoming data are generated and printed automatically on weekday mornings. The plotting scripts, which can be different for each experiment, are found with the processed data in the experiment subdirectories, currently on the workstation Puli. The program Plot Plus, written by Donald Denbo of NOAA's Pacific Marine Environmental Laboratory, is used to create multi-panel plots of instrument data. Maps of mooring locations are created using Matlab, a commercial numerical software package.

Plots of Argos data are used to monitor both meteorology and instrumentation, so we display the variables of meteorological interest on a separate plot from those of engineering interest. The scripts that create these plots are described here, see Appendix VII for examples.

8.1 Plot Plus Time Series Plots

8.1.1 Scientific Variables Plot

The command file argoss.ppc is used with program pplus to plot VAWR Argos meteorological data. The arguments are start day, end day, and input filename. The automated processing system invokes this script with a 3-day time span ending on the current day provided by the UNIX date utility.

See Appendix VII.2 for the text of the script argoss.ppc and a sample output plot.

8.1.2 Engineering Variables Plot

The pplus command file argose.ppc plots the variables that are of interest primarily for engineering purposes. Some of these variables are reported by the VAWR, some by the IMET or the tension recorder, depending on the Argos configuration for each experiment. The arguments are identical to the arguments for argoss.ppc, above. See Appendix VII.3 for the text of argose.ppc and a sample output plot.

8.2 Matlab Position Plot

Matlab script posit1.m is used to plot Argos positions. Because it is essential to see the most recent position value, we allow the plot to self-scale. This assures that any change in position will be very visible on the final plot, not cropped because it is out-of-bounds.

For some experiments, a watch circle is overplotted on the positions. This indicates the calculated limits of the mooring line, and is based on the depth of the mooring. Care must be used when plotting watch circles, however, since the accuracy of the Argos position information is only about one kilometer. For deep-water moorings, like the Arabian Sea, this is sufficient for the large watch circle of over 3 km. For a shallow mooring like GLOBEC, with its watch circle of about 40 meters, the Argos error exceeds the expected watch circle, so the watch circle is not plotted.

See Appendix VII.4 for the text of the Matlab position script, and sample output plots.

9 Appendices

Appendix I. Startup Shell Scripts

Appendix I.1. Shell Script argos.sh

```
#!/bin/sh
#
# @(#)argos.sh
#  /gdata/argos/argos.sh
#
# usage: sh argos.sh
#
# 931005 n.galbraith
#
# Tests for presence of file before archiving and processing
# Incoming data is moved to argos(2).dat.n where n is
# a constantly incrementing number.
#
# also processes engineering data (tension)
# uses findlast shell script in scripts subdirectory
# normally invoked by cron. use crontab -l to check.
#
# Be Careful Editing This!!!!!
# minor mod 940725 to send stdout of gargle.sh to logfile,not mail
# 940908 remove met1 (the pc) from the loop
# 940930 redesign: all processing done here, plotting done last

if [ -f /gdata/argos/argos.dat ];
then
        if [ -f /gdata/argos/argos.dat.1 ];
        then
                lastf=`/gdata/argos/scripts/findlast.sh ``/gdata/argos/argos.dat'''
                nextf=`expr $lastf + 1`
                date >> /gdata/argos/argos.log
                echo " moving argos.dat argos.dat.$nextf"\
                   >> /gdata/argos/argos.log
```

```
                mv /gdata/argos/argos.dat \
                           /gdata/argos/argos.dat.$nextf
        else
                nextf=1
                date >> /gdata/argos/argos.log
                echo " moving argos.dat argos.dat.1"\
                   >> /gdata/argos/argos.log
                mv /gdata/argos/argos.dat /gdata/argos/argos.dat.1
        fi

        # process primary arabian sea I VAWR
        /gdata/argos/gargle.sh gargle argos.dat.$nextf arab vawr0721\
           >> /gdata/argos/vawr0721.log 2>&1

        # send primary VAWR data to navy
        /puli/data/arab/argos/domail.sh 0721

        # process tension
        /puli/data/arab/argos/dotens.sh  $nextf

        # process position
        /puli/data/arab/argos/posit/doposit.sh  $nextf

        # process primary arabian sea II VAWR
        /gdata/argos/gargle.sh garglenew argos.dat.$nextf arab2 \
           vawr0720  >> /gdata/argos/vawr0720.log 2>&1

        # process arabian sea II tension
        /puli/data/arab2/argos/dotens.sh  $nextf

        # process position
        /puli/data/arab2/argos/posit/doposit.sh  $nextf
fi

# now process the GLOBEC data in argos2.dat
if [ -f /gdata/argos/argos2.dat ];
then
        if [ -f /gdata/argos/argos2.dat.1 ];
        then
```

```
                lastf=`/gdata/argos/scripts/findlast.sh ``/gdata/argos/argos2.dat'''
                nextf=`expr $lastf + 1`
                date >> /gdata/argos/argos.log
                echo " moving argos2.dat argos2.dat.$nextf" \
                   >> /gdata/argos/argos.log
                mv /gdata/argos/argos2.dat \
                   /gdata/argos/argos2.dat.$nextf
        else
                nextf=1
                date >> /gdata/argos/argos.log
                echo " moving argos2.dat argos2.dat.1"\
                   >> /gdata/argos/argos.log
                mv /gdata/argos/argos2.dat /gdata/argos/argos2.dat.1
        fi

        # process GLOBEC primary VAWR from file argos2.dat.n
        /gdata/argos/gargle.sh garglenew argos2.dat.$nextf \
           globec vawr0707  >> /gdata/argos/vawr0707.log 2>&1

        # process GLOBEC primary IMET - includes tension
        /gdata/argos/imet.sh argos2.dat.$nextf globec imet1 \
           >> /gdata/argos/globecimet1.log 2>&1
        # process GLOBEC secondary VAWR
        /gdata/argos/gargle.sh garglenew argos2.dat.$nextf \
           globec vawr0380  >> /gdata/argos/vawr0380.log 2>&1

        # process GLOBEC secondary IMET
        /gdata/argos/imet.sh argos2.dat.$nextf globec imet2 \
           >> /gdata/argos/globecimet2.log 2>&1
        # send GLOBEC data to recipients
        /gdata/argos/globec/ftp.sh >> /gdata/argos/globec/ftp.log 2>&1

        # process position
        /puli/data/globec/argos/posit/doposit.sh  $nextf
fi
```

Appendix I.2. Shell Script Findlast.sh

This script works with groups of files with three-part names,where the parts of the names are separated by periods, and the third part of the name is a version number. It will find the file with name matching the two parts specified in argument 1 with the highest version number on disk. Findlast is called by argos.sh and doposit.sh, but it can also be invoked from the keyboard to determine the number of the most recent argos.dat or argos2.dat file. It returns only the number of the file, not the complete filename.

```
#!/bin/sh
#  find the files in the argument and return the number of highest
#  version number.
#  based on filenames file.ext.1 file.ext.2 .... file.ext.n
#
#  modified 950124 to handle deleted data files by using the version
#  number instead of relying on creation date of file
#
 if [ $# -eq 0 ]
 then
    echo -1
    exit
 fi
 ls $1* |awk 'BEGIN {FS="."}{print $3}' |sort +0 -n |tail -1
```

Appendix II. VAWR Processing

Appendix II.1. Shell Script gargle.sh

```
#!/bin/sh
# gargle.sh: run one of the gargle programs and process the output
# usage:            gargle.sh  program argosfilename experiment vawrname
# example:          gargle.sh garglenew argos2.dat.1 globec vawr0707
# 941208 n. galbraith
# arguments:
#   $1 program to run   garglenew         must be in /gdata/argos/src
#   $2 inputfile        argos2.dat.1      must be in /gdata/argos
#   $3 experiment       globec            subdirectory/eperiment
#   $4 vawrnam          vawr0707          table file in subdir named in $3
cd /gdata/argos                         # go to incoming data spot
src/$1 $2 $3/$4.tbl  $3/$4.raw          # run garglenew or gargle
sort +0 -1 -u -n $3/$4.raw > $3/$4.asc  # remove incoming dupes
cat $3/$4.asc >> $3/$4.arch             # archive, remove duplicates and
                                        # send to processing dir
sort +0 -1 -u -n $3/$4.arch > /puli/data/$3/argos/$4.asc
```

Appendix II.2. Gargle output

This is a sample output file containing VAWR data as written by program gargle. The variables are:

```
yearday east north speed dir sw  rh  bp     seaT   airT TPV  bodyT domeT lw
        m/s   m/s   m/s  deg w/m-2 % mb     degC   degC mv   DegK  DegK  w/m-2
52.229,-4.79,1.66, 5.07,25.8,-1,95.4,995.6, 5.552, 6.74,41.1,279.8,279.8,352.8
52.250,-3.78,2.41, 4.48,37.5,-1,95.5,995.0, 5.559, 6.59,42.9,279.7,279.6,352.6
52.260,-3.37,2.86, 4.42,29.6,-1,95.6,994.9, 5.558, 6.57,44.4,279.7,279.8,352.9
52.281,-2.70,4.11, 4.92,13.3,-1,95.7,993.9, 5.546, 6.68,48.0,279.8,279.7,354.4
```

Appendix II.3. VAWR Table File

```
-> vawr_tabl :0721 : nan : Tue Jun 28 94
#0 :v721wr    instrument identifier
           01356 - PROGRAM
           06856 - PTT
           06857 - PTT
           06858 - PTT
#1 :EC  east counts
              16 - BITS_EC
           -10.0 - MIN_EC
            30.0 - MAX_EC
           -99.0 - MISS_EC
#2 :NC  north counts
              16 - BITS_NC
           -10.0 - MIN_NC
            30.0 - MAX_NC
           -99.0 - MISS_NC
#3 :RC rotor counts
              16 - BITS_RC
           -10.0 - MIN_RC
            30.0 - MAX_RC
           -99.0 - MISS_RC
#4 :CO CO counts
               8 - BITS_CO
           -10.0 - MIN_CO
            30.0 - MAX_CO
           -99.0 - MISS_CO
             -.8 - MAGVAR
#5 :VA vane counts
               8 - BITS_VA
           -10.0 - MIN_VA
            30.0 - MAX_VA
           -99.0 - MISS_VA
#6 :TM time
```

```
              16 - BITS_TM
        19920101 - MIN_TM
        19941231 - MAX_TM
           -99.0 - MISS_TM
             450 - TSAMP
             450 - CWIS
              94 - YEAR
              10 - MONTH
              02 - DAY
              06 - HOUR
              30 - MINUTE
              00 - SECOND
#7 :SW short wave radiation
              20 - BITS_SW
           -10.0 - MIN_SW
          1500.0 - MAX_SW
           -99.0 - MISS_SW
           25418 - #_SW
           10.52 - CAL_SW
             4.0 _ Z_SW
#8 :LW long wave radiation
              20 - BITS_LW
           100.0 - MIN_LW
           500.0 - MAX_LW
           -99.0 - MISS_LW
           28463 - #_TP
         2.04164 - LW_A
            1.11 - LW_B
              0. - LW_C
       -1018.68 - TP_A
         2.04164 - TP_B
#9 :RH relative humidity
              16 - BITS_RH
             2.0 - MIN_RH
           115.0 - MAX_RH
           -99.0 - MISS_RH
             034 - #_RH
```

```
          5.0667 - A_RH
         0.04571 - B_RH
      -0.0012539 - C_RH
         3.51563 - TI_RH
          210.32 - P20
             20. - P20FACT
#10 :BP  barometric pressure
              16 - BITS_BP
           800.0 - MIN_BP
          1100.0 - MAX_BP
           -99.0 - MISS_BP
           43698 - #_BP
             0.0 - A_BP
             0.0 - B_BP
        92.68879 - C1_BP
        2.672706 - C2_BP
       -114.7084 - C3_BP
        0.031184 - D1_BP
   27.92331E-06 - T0_BP
        2.636716 - TI_BP
        27.85152 - T1_BP
        0.831612 - T2_BP
        20.34282 - T3_BP
       -4035.285 - Y1_BP
             0.0 - Y1_BP
      -14001.10 - Y2_BP
#11 :ST sea temperature
              20 - BITS_SEA
           -10.0 - MIN_SEA
            40.0 - MAX_SEA
           -99.0 - MISS_SEA
            5005 - #_SEA
         9038.16 - R1_SEA
         3998.49 - R2_SEA
           -1.45 - F1_SEA
          759.14 - F2_SEA
 .109376436E-02 - TA_SEA
```

```
  .262474576E-03 - TB_SEA
  .146244473E-06 - TC_SEA
         3608.0 - RS_SEA
-> number of multiplexed temperature
-> sensors in the instrument
            4.0 - K_TEMP
#12 :AT air temperature
             20 - BITS_AIR
          -20.0 - MIN_AIR
           40.0 - MAX_AIR
          -99.0 - MISS_AIR
           5804 - #_AIR
        12699.00 - R1_AIR
         4999.70 - R2_AIR
         -131.18 - F1_AIR
          711.50 - F2_AIR
  .128804283E-02 - TA_AIR
  .235519671E-03 - TB_AIR
  .951705804E-07 - TC_AIR
         4296.0 - RS_AIR
#13 :BT  body temperature
             20 - BITS_B
          -10.0 - MIN_B
           50.0 - MAX_B
          -99.0 - MISS_B
          284631 - #_LWB
         23460.0 - R1_B
         10000.0 - R2_B
          -65.50 - F1_B
          712.40 - F2_B
   .10125311E-02 - TA_B
   .24220731E-03 - TB_B
   .14217244E-06 - TC_B
         4465.0 - RS_B
#14 :DT  dome temperature
             20 - BITS_D
          -20.0 - MIN_D
```

```
            50.0 - MAX_d
           -99.0 - MISS_D
          284632 - #_LWD
         23460.0 - R1_D
         10000.0 - R2_D
          -66.56 - F1_D
          711.49 - F2_D
   .10161020E-02 - TA_D
   .24160036E-03 - TB_D
   .14344824E-06 - TC_D
         4450.0 - RS_D
#15 :DU filler engineering byte
               8 - BITS_DUM
#16 :CS checksum
              16 - BITS_CHECKSUM
->    /* must be 1st non standard const */
              30 - LST_BYT
->    /* must be 2nd const,0-bin 1-asc  */
               1 - ASCII
```

Appendix III. IMET Processing

Appendix III.1. Shell Script Imet.sh

```
#!/bin/sh
# imet.sh   runs gargle_imet
#
# usage:
# imet.sh argosfilename directory/experiment instrument
#       where argosfilename is the raw incoming argos file
#             experiment   is the directory experiement i.e. globec
#             instrument_name is normally imet1 or imet2
#
# 941208 n. galbraith
#
cd /gdata/argos                         # go to incoming data spot
src/gargle_imet $1  $2/$3.tbl  $2/$3.raw
tr ',' ' ' < $2/$3.raw | awk -f imet.awk  |sort +0 -1 -u -n > \
             $2/$3.asc                  # remove incoming dupes
cat $2/$3.asc >> $2/$3.arch             # archive
                                        # remove dupes from whole set
                                        # and send to processing dir
sort +0 -1 -u -n $2/$3.arch > /puli/data/$2/argos/$3.asc
```

Appendix III.2. Imet.awk

```
BEGIN {
pday = "date +%j";
NUL = 0;
}
{       hour =  $2 + .5;
jday = $1 + (hour / 24.);
if ($1 <= pday)  {
          at = $4;
          if (jday < 91.)  {
if (at > 25. )  {
at = at - 35.96 - 5.;
}
  }
  printf("%9.4f,%9.4f,%9.4f,%2d,%2d,%9.4f,",jday,$6,$7,NUL,NUL,$9)
  printf("%9.4f,%9.4f,%9.4f,%9.4f,%2d,%2d,%2d,",$8,$3,at,$5,NUL,NUL,NUL);
  printf("%9.4f,%9.4f,%9.4f,%3d\n",$10,$12,$13,$2)
}
else
  print $1, jday, pday, tday
}
# process argos imet
# input variables:
# day hr bp at st u v rh sw lw rn bv tens
# output variables:
# realday u v null null sw rh bp at st null null null lw bv tens hour
# this version 950104 checks for clock jumping ahead of today's date
# 950202    added 30 minutes to time to reflect center point of record
# 950204    normalize very low temperatures
# air temp   max = 35.95   min = -5.
# sea temp   max = 38.95   min = -2.
# to normalize ''wrapped'' values, subract (max val + .01), add min value
# i.e.  at val  = 33.5, actual val should be 33.5 - 35.95 - 5. = -7.4
# for this experiment, use max acceptible val for at = 25
```

Appendix III.3 Sample IMET File

Variables labelled d are dummy variables.

```
yearday    u     v    d d sw    rh    bp      airT seaT d d d lw     bv  tns   prc
          m/s   m/s      w/m^2  %     mb      degc degc       w/m^2 v   lbs   mm
53.1042,-5.00,5.30,0,0,0.0,91.0, 995.8,3.10, 5.46,0,0,0,321.0,13.,3450.,0
53.1458,-4.10,7.00,0,0,0.0,90.0, 996.0,2.95, 5.46,0,0,0,322.0,13.,3350.,9
53.1875,-3.30,8.20,0,0,0.0,91.0, 996.2,2.55, 5.53,0,0,0,320.0,13.,3300.,9
53.2292,-2.50,7.80,0,0,0.0,90.0, 996.4,2.48, 5.56,0,0,0,317.0,13.,3300.,13
53.2708,-3.30,8.00,0,0,0.0,88.0, 996.8,2.46, 5.63,0,0,0,313.0,13.,3050.,13
53.3125,-4.50,7.90,0,0,0.0,87.0, 996.8,2.35, 5.60,0,0,0,313.0,13.,3000.,0
```

Appendix III.4. IMET Table File

```
-> imet_tabl :01419 : roger : Thu Dec 08 95
#0 :i01419    instrument identifier
           01419 - PROGRAM
           23657 - PTT
           23658 - PTT
           23659 - PTT
#1 :DA  day of year
              09 - BITS_DA
             1.0 - MIN_DA
           365.0 - MAX_DA
           -99.0 - MISS_DA
#2 :HR  hour of day
              05 - BITS_HR
             0.0 - MIN_HR
            23.0 - MAX_HR
           -99.0 - MISS_HR
#3 :BP  barometric pressure
              11 - BITS_BP
           900.0 - MIN_BP
          1100.0 - MAX_BP
           -99.0 - MISS_BP
#4 :AT  air temperature
              12 - BITS_AT
            -5.0 - MIN_AT
            35.0 - MAX_AT
           -99.0 - MISS_AT
            0.01 - SCALE_AT
           500.0 - BIAS_AT
#5 :WT  sea temperature
              12 - BITS_WT
            -2.0 - MIN_WT
            38.0 - MAX_WT
           -99.0 - MISS_WT
            0.01 - SCALE_WT
```

```
          200.0 - BIAS_WT
#6 :WU  Wind U
             10 - BITS_WU
          -51.5 - MIN_WU
           51.5 - MAX_WU
          -99.0 - MISS_WU
            0.1 - SCALE_WU
          511.0 - BIAS_WU
#7 :WV  Wind V
             10 - BITS_WV
          -51.5 - MIN_WV
           51.5 - MAX_WV
          -99.0 - MISS_WV
            0.1 - SCALE_WV
          511.0 - BIAS_WV
#8 :RH  relative humidity
              7 - BITS_RH
            0.0 - MIN_RH
          100.0 - MAX_RH
          -99.0 - MISS_RH
#9 :SW  short wave radiation
             11 - BITS_SW
            0.0 - MIN_SW
         1500.0 - MAX_SW
          -99.0 - MISS_SW
#10 :LW long wave radiation
             10 - BITS_LW
            0.0 - MIN_LW
          750.0 - MAX_LW
          -99.0 - MISS_LW
#11 :RN  precipitation
              7 - BITS_RN
            0.0 - MIN_RN
          127.0 - MAX_RN
          -99.0 - MISS_RN
#12 :BV  battery voltage
              8 - BITS_BV
```

```
          -20.0 - MIN_BV
           15.0 - MAX_BV
          -99.0 - MISS_BV
            0.1 - SCALE_BV
#13 :TN  tension
               8 - BITS_TN
          -20.0 - MIN_TN
        12750.0 - MAX_TN
          -99.0 - MISS_TN
           50.0 - SCALE_TN
-> NOTE: the cycle repeats, identical values
#14 :D2  day of year
              09 - BITS_DA
             1.0 - MIN_DA
           365.0 - MAX_DA
          -99.0 - MISS_DA
#15 :H2  hour of day
              05 - BITS_HR
             0.0 - MIN_HR
            23.0 - MAX_HR
          -99.0 - MISS_HR
#16 :B2  barometric pressure
              11 - BITS_BP
           900.0 - MIN_BP
          1100.0 - MAX_BP
          -99.0 - MISS_BP
#17 :A2  air temperature
              12 - BITS_AT
            -5.0 - MIN_AT
            35.0 - MAX_AT
          -99.0 - MISS_AT
            0.01 - SCALE_AT
          500.0 - BIAS_AT
#18 :W2  sea temperature
              12 - BITS_WT
            -2.0 - MIN_WT
            38.0 - MAX_WT
```

```
          -99.0 - MISS_WT
           0.01 - SCALE_WT
          200.0 - BIAS_WT
#19 :U2  Wind U
             10 - BITS_WU
          -51.1 - MIN_WU
           51.1 - MAX_WU
          -99.0 - MISS_WU
            0.1 - SCALE_WU
          511.0 - BIAS_WU
#20 :V2  Wind V
             10 - BITS_WV
          -51.1 - MIN_WV
           51.1 - MAX_WV
          -99.0 - MISS_WV
            0.1 - SCALE_WV
          511.0 - BIAS_WV
#21 :R2  relative humidity
              7 - BITS_RH
            0.0 - MIN_RH
          100.0 - MAX_RH
          -99.0 - MISS_RH
#22 :S2  short wave radiation
             11 - BITS_SW
            0.0 - MIN_SW
         1500.0 - MAX_SW
          -99.0 - MISS_SW
#23 :L2 long wave radiation
             10 - BITS_LW
            0.0 - MIN_LW
          750.0 - MAX_LW
          -99.0 - MISS_LW
#24 :P2  precipitation
              7 - BITS_RN
            0.0 - MIN_RN
          127.0 - MAX_RN
          -99.0 - MISS_RN
```

```
#25 :C2  battery voltage
                8 - BITS_BV
            -20.0 - MIN_BV
             15.0 - MAX_BV
            -99.0 - MISS_BV
              0.1 - SCALE_BV
#26 :T2  tension
                8 - BITS_TN
            -20.0 - MIN_TN
          12750.0 - MAX_TN
            -99.0 - MISS_TN
             50.0 - SCALE_TN
#27 :CS checksum
               16 - BITS_CHECKSUM
->             /* must be 1st non standard constant */
               30 - LST_BYT
->             /* must be 2nd constant 0-bin 1-asc  */
                1 - ASCII
```

Appendix IV. Engineering PTT Processing

Appendix IV.1. Shell Script dotens.sh

```
#!/bin/sh
# process tension, and battery voltage for arabian sea
# the engineering ptt is # 22527.

#  Data characters are
#     AA BB CC DD EE FF GG HH
#  where:
#     BB = battery voltage
#     CC = tension
#
#  Tension:
#    T = (CCd/256) * 5.0 * 2067.9  - 423.0
#    where:
#        AAd is the decimal equivalent of AA hex.
#
#  Battery voltage:
#     BV = (dBB/256) * 17.1

# sample input Argos records:
# 01356 22527 3 32 H 1 1994-06-01 10:45:24 41.533 289.356 0.0 401649554
#  1994-06-01 10:49:10 4 E3 C6 18 99
#  00 00 00 FF
# 01356 22527 5 32 D 2 1994-06-01 11:09:54 41.528 289.356 0.0 401649554
#  1994-06-01 11:06:08 2 E3 C6 18 99
#  00 00 00 FF
#  1994-06-01 11:15:33 3 E3 C5 18 99
#  00 00 00 FF
#
# output file tens.asc:
#  Doy,        Yy,  Mo,Da,Hr,Mn, Sc, Tens,     Dummy, Batt Volt
# 155.959625, 1994, 6, 4, 23, 1, 52, 425.162109, 000, 13.158984
```

```
# to remove data from the database, edit file tens.raw, which is
# appended by this script
# if an argument is used, it is the number of the file to process
# if called with no arguments, process most recent file
if [ $# -eq 1 ]
then
       lastf=$1
elif [ -f /gdata/argos/argos.dat.1 ]
then
       lastf=`/gdata/argos/scripts/findlast.sh ``argos.dat.*'''
else
       lastf=1
fi

if [ ! -f /puli/data/arab/argos/tens.raw ]
then
       touch  /puli/data/arab/argos/tens.raw
fi

# extract tens data from the Argos file and append
#         to the raw tens file
tr ':\-' ' '<  /gdata/argos/argos.dat.$lastf | awk -f \
   /puli/data/arab/argos/tens.awk >>  /puli/data/arab/argos/tens.raw

sort +0 -n -u <  /puli/data/arab/argos/tens.raw | \
   /puli/data/arab/argos/conv >  /puli/data/arab/argos/tens.asc
chmod 777  /puli/data/arab/argos/tens.raw
chmod 777  /puli/data/arab/argos/tens.asc
```

Appendix IV.2. Awk Script tens.awk

```
BEGIN {start=0}
$2 ~ /22527/ {start=1}
$1 ~ /1356/ { if ($2 !~ /22527/) {start = 0}}
$2 !~ /22527/ { if (start)
      { if (NF == 11) {
 year = $1;
         month = $2;
         day = $3;
         hour = $4;
         minute = $5;
         second = $6;
         jhour =  hour/24 + minute/24/60 + second/24/60/60;
         doy = day;
         if (month == 1)
       {doy =  day }
         if (month == 2)
       {doy =  31 + day }
         if (month == 3)
       {doy =  59 + day }
         if (month == 4)
       {doy = 90 + day }
         if (month == 5)
       {doy = 120 + day }
         if (month == 6)
       {doy = 151 + day }
         if (month == 7)
       {doy = 181 + day }
         if (month == 8)
       {doy = 212 + day }
         if (month == 9)
       {doy = 243 + day }
         if (month == 10)
           {doy =  273 + day }
         if (month == 11)
```

```
        {doy =  304 + day }
          if (month == 12)
        {doy =  334 + day }
          doy = doy + jhour;

printf ("%f %d %d %d %d %d %s %s %s\n",
   doy,$1,$2,$3,$4,$5,$6,$9,$10,$11)}
}
}
```

Appendix IV.3. Program Conv.c

This program is used by the shell script dotens.sh to decode the ASCII hexadecimal values in the tensiometer PTT record.

```
#include <stdio.h>
#include <string.h>

main ()

{
  int ii, inb,outb, stat;
  int yy,mo,da,hr,mn,sc,tn,tt,bat;
  float BV,T,doy;
  while(1)
     {
     stat=scanf( "%f %d %d %d %d %d %x %x %x",
               &doy,&yy,&mo,&da,&hr,&mn,&sc,&bat,&tn,&tt);
     if(stat == EOF)
               exit(0);
     T  = (tn/256.) * 5.0 * 2067.9  - 423.0;
     if (T < 0.5) T = 0;
     BV = (bat/256.) * 17.1 ;

     printf( "%f, %d, %d, %d, %d, %d, %d, %f, %d, %f\n" ,
       doy,yy,mo,da,hr,mn,sc,T,tt,BV);
   }
}
```

Appendix V. Position Processing Scripts

Appendix V.1. doposit.sh

```
#! /bin/sh
#
#       process Argos position data for Arabian Sea experiment
#       94/03/02 N. Galbraith
# usage:
# doposit.sh (with no arguments)
#      processes most recent file in /gdata/argos/argos.dat.n
# doposit.sh nn
#      processes file /gdata/argos/argos.dat.nn
# argos.dat:
# 01356 06856 33 32 D 1 1994-05-20 12:06:37 \
#            41.535 289.353 0.002 401650137

cd /puli/data/arab/argos/posit

# find the input file. If not specified on command line,
#    process the newest argos.dat.n file
if [ $# -eq 1 ]
then
       lastf=$1
elif [ -f /gdata/argos/argos.dat.1 ]
then
       lastf=`/gdata/argos/scripts/findlast ``argos.dat'''
else
       lastf=1
fi

# extract position info from Argos file
tr ':\-' ' ' < /gdata/argos/argos.dat.$lastf |\
    awk 'NF > 13 {print $0}'|  awk -f posit.awk > junk.dat

if [ ! -f  pripos.arch.asc ]
```

```
then
       touch  pripos.arch.asc
fi

np='awk '$1 == 6856' junk.dat |wc -l'
if [ $np -ge 1 ]
then
  echo processing primary position reporter
  awk '$1 == 6856 { print $2, $3, $4, $5, $6, $7, $8, $9, $10 }'\
        junk.dat >> pripos.arch.asc
  cat  pripos.arch.asc | sort -n +6 -u > arab1.pos
fi

np='awk '$1 == 6859' junk.dat |wc -l'
if [ $np -ge 1 ]
then
  echo processing spare position reporter
  awk '$1 == 6859 { print $2, $3, $4, $5, $6, $7, $8, $9, $10 }' \
       junk.dat >> secpos.arch.asc
 cat  secpos.arch.asc | sort -n +6 -u > arab2.pos
fi
```

Appendix V.2. posit.awk

```
{     year = $7;
    if (year > 1900) {year = year-1900 }
    month = $8;
    day = $9;
    hour = $10;
    minute = $11;
    second = $12;
       jhour =  hour/24 + minute/24/60 + second/24/60/60;
            doy = day;
            if (month == 2) {doy = 31 + day }
            if (month == 3) {doy = 59 + day }
            if (month == 4) {doy = 90 + day }
            if (month == 5) {doy = 120 + day }
            if (month == 6) {doy = 151 + day }
            if (month == 7) {doy = 181 + day }
            if (month == 8) {doy = 212 + day }
            if (month == 9) {doy = 243 + day }
    if (month == 10)    {doy =  273 + day }
    if (month == 11)  {doy =  304 + day }
    if (month == 12)  {doy =  334 + day }
    doy = doy + jhour;
    long = $14;
    if (long > 180)
        {long = $14 - 360 }
            printf("%d %2d %2d %2d %2d %2d %2d %7.3f %8.3f %8.3f\n",
               $2,year,month,day,hour,minute,second,doy,$13,long)}{}
```

Appendix VI. Data Distribution Scripts

Appendix VI.1. Ftp Shell Script

```
#!/bin/sh
#    sends 4 files to anonymous ftp site
#    note: names and addresses have been changed for publication
#
ftp -n -i  128.128.99.99 << EOF
user anonymous myname@myhost.myorganization.mydomain
cd incoming/globec.burnin
ascii
put /puli/data/globec/argos/vawr0707.asc  vawr0707.asc
put /puli/data/globec/argos/vawr0380.asc vawr0380.asc
put /puli/data/globec/argos/imet1.asc imet1.asc
put /puli/data/globec/argos/imet2.asc imet2.asc
bye
EOF
```

VI.2. Mail Shell Script

```
#!/bin/sh
# usage domail.sh xxxx where xxxx is 4-dgit VAWR number
#
# 941114 nrg    - for Arabian Sea I
#
# input:  file vawrxxxx.asc
# output: file  nxxxx.asc
#
cd /puli/data/arab/argos

# write the header to output file
echo " WHOI Arabian Sea Met Data - Robert A. Weller" > n$1.asc
echo "  day-time  wspd   wdir    sr  rh   bp     sst   at    lr  "\
       >> n$1.asc

# write the last 100 points to the output file
tail -100 vawr$1.asc | tr ',' ' '|awk -f x.awk >> n$1.asc

# send the file somewhere here. Use rcp or mail
mail -s "WHOI Arabian Sea Met Data" user1@comp1.navy.mil < n$1.asc
```

Appendix VI.3. x.awk

This one-line script extracts data variables of interest to be transmitted to remote sites.

```
{ printf("%10.6f %5.2f %7.2f %4d %4.1f %6.1f %5.2f %5.2f %5.1f\n",
$1,$4,$5,$6,$7,$8,$9,$10,$14)}
```

Appendix VII. Plotting Scripts and Sample Plots

Appendix VII.1. Shell Script dopl.sh

Shell script dopl.sh plots VAWR and mooring information.

```
#!/bin/sh
# 940930 N. Galbraith
# plot Argos data for arabian sea deployment 1
#               VAWR science variables
#               VAWR engineering & other engineering variables
#               position, using posit/posit1.m
#
# set up to run pplus
LD_LIBRARY_PATH=/usr/openwin/lib                # environment
export LD_LIBRARY_PATH
cd /puli/data/arab/argos

ed=`date "+%j"`                                 # date limits
sd=`expr $ed - 3`

if [ -f ppl.meta001 ]                           # housekeeping
then
       rm ppl.meta*
fi
                                                # do it
# plot the science and engineering variables for primary VAWR
/usr/local/bin/pplus argoss.ppc $sd $ed vawr0721.asc
/usr/local/bin/pplus argose.ppc $sd $ed vawr0721.asc

chmod 666 ppl.meta001
chmod 666 ppl.meta002

# plot the positions with Matlab script posit1.m
# which creates plot file pposit.ps
```

```
cd /puli/data/arab/argos/posit
/usr/local/bin/matlab <posit1.m
chmod 666 pposit.ps

# send the plotfiles to the printer
cd /puli/data/arab/argos
/usr/local/bin/m2ps -R ppl.meta001 | lpr
if [ -f ppl.meta002 ]
then
       /usr/local/bin/m2ps -R ppl.meta002 | lpr
fi

cd /puli/data/arab/argos/posit
lpr pposit.ps
```

VII.2. Pplus Plotting Scripts

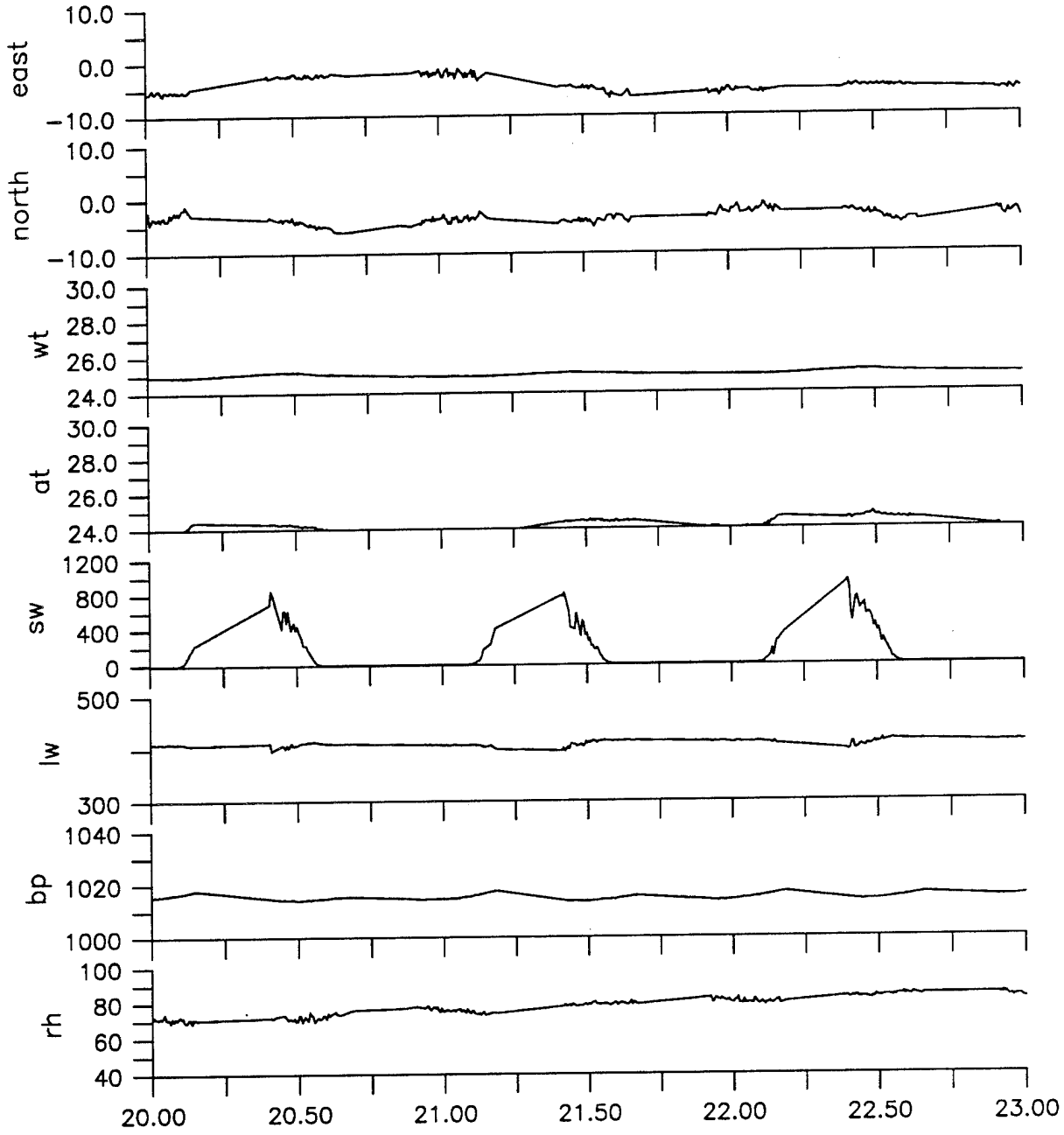
VII.2.1. Plot Scientific Variables: argoss.ppc

```
c pplus plotting script argoss.ppc
c
c usage:
c     pplus argos.ppc styrday enyrday file
c     note plot type hard wired to 0 for file output

pltype,0
axset,0,1,1,0
rotate on
window on

multplt 1,8
5.75
0.7,0.7,0.7,0.7,0.7,0.7,0.7,0.7
1.5
0.2,0.2,0.2,0.2,0.2,0.2,0.2,1.5

line 1,,0
c  set up the year day for the xaxis
xaxis 'p1','p2',.25
axlabp 0,-1

yaxis,-10,10,5,
format free
vars 1,1,2
skp,2,'p3'
rd
ylab east
plot,

yaxis,-10,10,5
format free
```

```
vars 1,1,,2
skp,2,'p3'
rd
ylab north
plot,

yaxis,10,40,10
format free
vars 1,1,,,,,,,2
skp,2,'p3'
rd
ylab wt
plot,

yaxis,10,40,10
format free
vars 1,1,,,,,,,,2
skp,2,'p3'
rd
ylab at
plot,

yaxis,-50,1200,200
yfor (i4)
format free
vars 1,1,,,,,2
skp,2,'p3'
rd
ylab sw
plot,

yaxis,0,600,200
yfor (i3)
format free
vars 1,1,,,,,,,,,,,,2
skp,2,'p3'
rd
```

```
ylab lw
plot,

yaxis,1000,1040,10,
yfor (i4)
format free
vars 1,1,,,,,,,2
skp,2,'p3'
rd
ylab,bp
plot,

axlabp -1,-1
yaxis,0,100,20
yfor (i3)
format free
vars 1,1,,,,,,2
skp,2,'p3'
rd
ylab,rh
plot,Arabian Sea Argos 'p3'
```

VII.2.2. Scientific Variables Plot



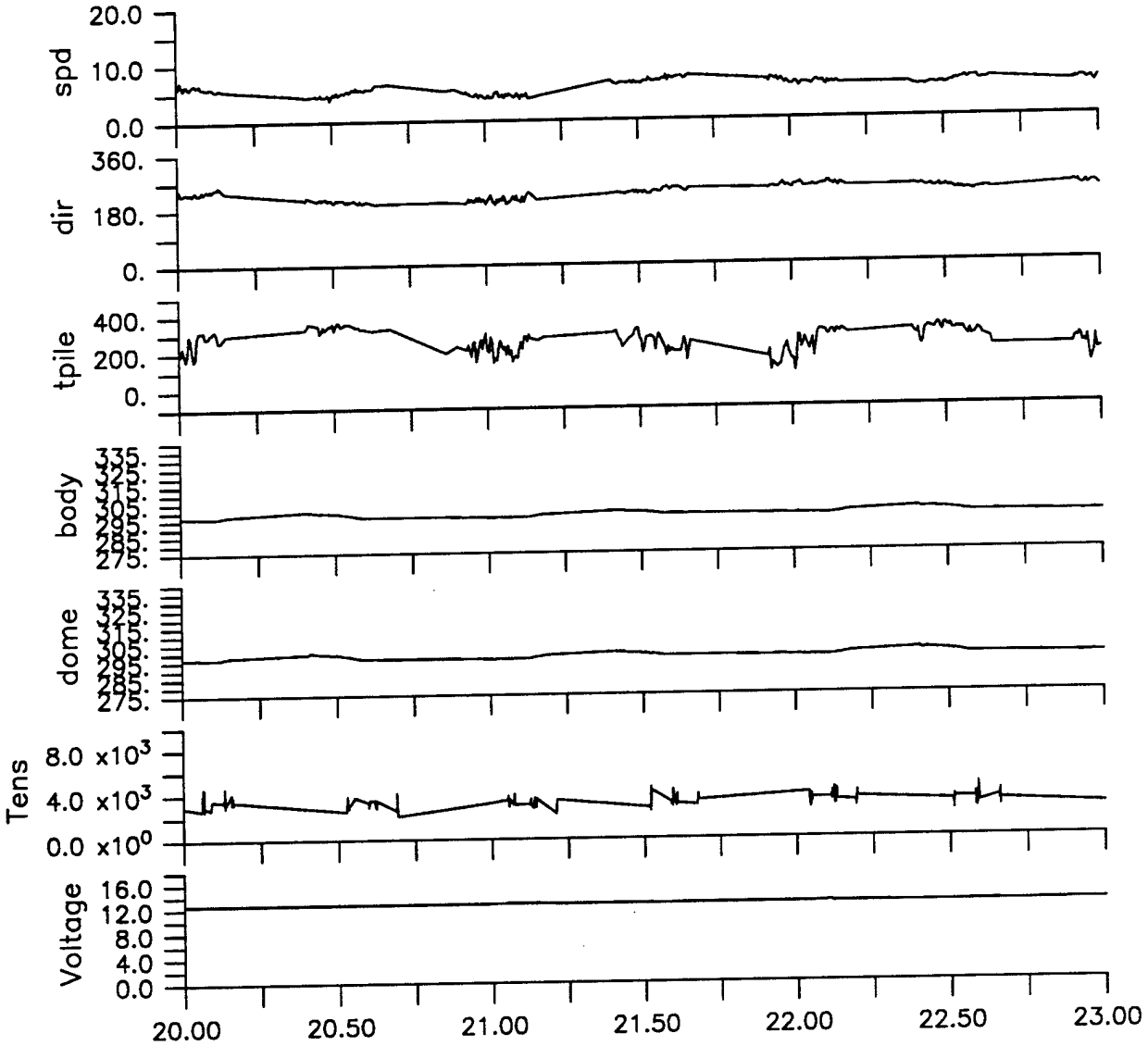
Arabian Sea Argos vawr0721.asc

VII.2.3. Plot Engineering Variables: argose.ppc

```
c plot VAWR Argos engineering data for Arabian Sea
c usage:  pplus argos.ppc styrday enyrday file
pltype,0
axset,0,1,1,0
multplt 1,7
5.75
0.7,0.7,0.7,0.7,0.7,0.7,0.7
1.5
0.2,0.2,0.2,0.2,0.2,0.2,1.5
line 1,,0
xaxis 'p1','p2',.25
window on
axlabp,0,-1

yaxis,0,10,1
format free
vars 1,1,,,2
skp,2,'p3'
rd
ylab,spd
plot,

yaxis,0,360,90
vars 1,1,,,,2
skp,2,'p3'
rd
ylab,dir
plot,

yaxis,-200,400,100
vars 1,1,,,,,,,,,2
skp,2,'p3'
rd
ylab,tpile
```

```
plot,

yaxis,275,340,5
vars 1,1,,,,,,,,,,,2
skp,2,'p3'
rd
ylab,body
plot,

yaxis,275,340,5
vars 1,1,,,,,,,,,,,2
skp,2,'p3'
rd
ylab,dome
plot,

yaxis,
vars 1,1,,,,,,,2
skp,2,tens.asc
rd
ylab,Tens
plot,

axlabp -1,-1
yaxis,
vars 1,1,,,,,,,,,2
skp,2,tens.asc
rd
ylab,Voltage
plot,Arabian Sea Argos 'p3'
```

VII.2.4. Engineering Variables Plot



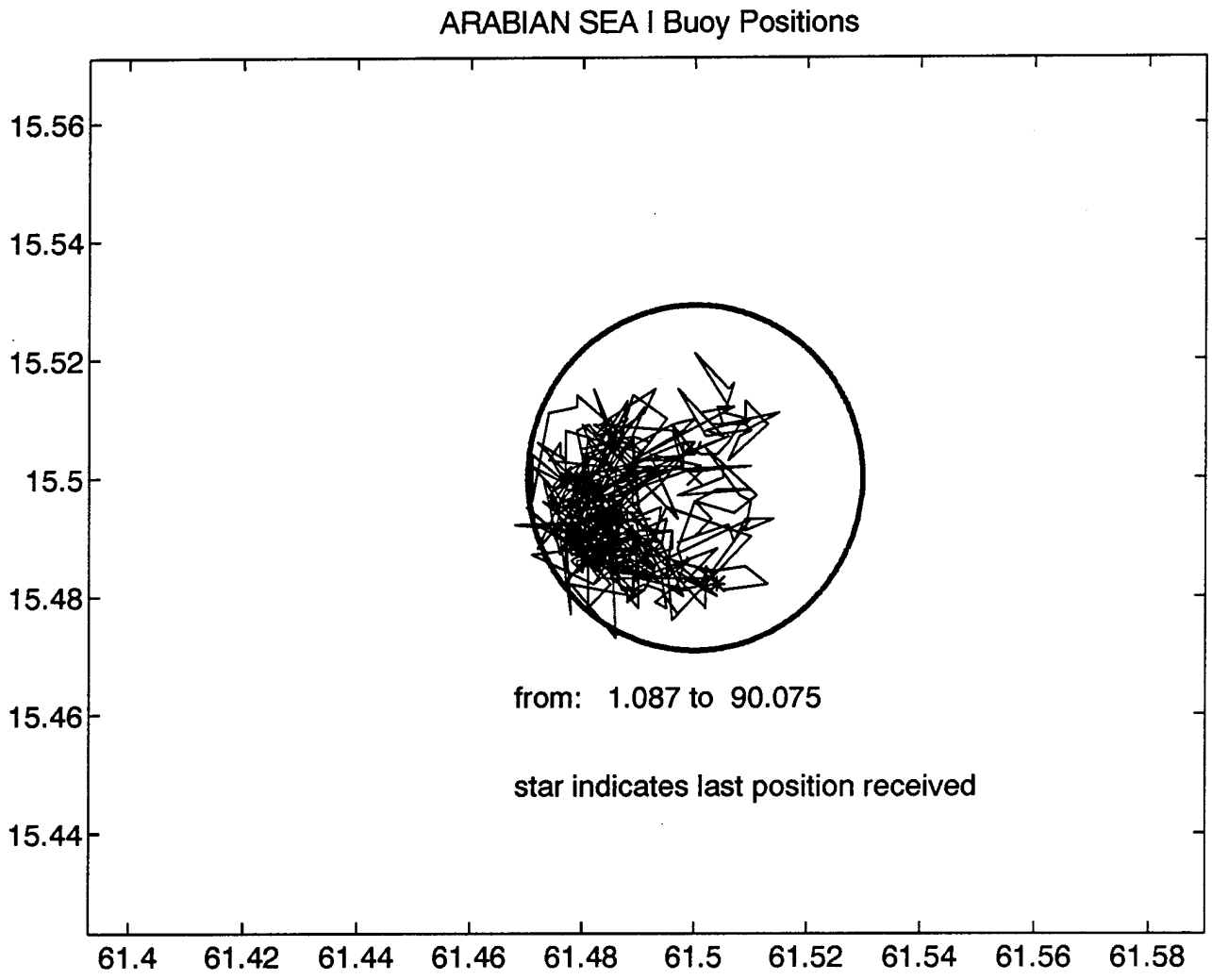
Arabian Sea ARGOS vawr0721.asc

VII.3. Matlab Position Plotting Argos position values are plotted using Matlab noninteractively with script posit1.m.

VII.3.1 Matlab Script

```
%   plot a position-only Argos file
%   941006 n.galbraith   - for Arabian Sea
load arab1.pos -ascii
plot(arab1(:,9),arab1(:,8))
hold on
%   get label position
minx = min(arab1(:,9));   miny = min(arab1(:,8));
maxx = max(arab1(:,9));   maxy = max(arab1(:,8));
axis([minx-.05,maxx+.05,miny-.05,maxy+.05])
%   get and annotate end date
[xi,yi] =size(arab1);
txt = sprintf('from: %7.3f to %7.3f',arab1(1,7),arab1(xi,7));
text(minx, miny, txt);
text(minx, miny-.025,'star indicates last position received');
plot(arab1(xi,9),arab1(xi,8),'*');
%   plot the watch circle and the anchor position
ay = 15.5;    ax = 61.5;
plot(ax,ay,'x')
rady =  1.75 / 60;    %  1.75 nm  % 60 nm = 1degree
radx = 1.75 / (cos(ay) * 60);
xcirc = zeros(360);  ycirc = zeros(360);
for ii=1:360
       xcirc(ii)=radx*(cos(ii))+ax;
       ycirc(ii)=rady*(sin(ii))+ay;
end
plot(xcirc,ycirc,'.')
title( 'ARABIAN SEA I Buoy Position');
print pposit.ps
```

VII.3.1 Position Plot



Acknowledgements

The author wishes to thank members of the Upper Ocean Processes Group for discussions and feedback during the development of the software. Discussions with N. Brink and A. Plueddemann during the preparation of this document are gratefully acknowledged.

DOCUMENT LIBRARY

Distribution List for Technical Report Exchange – May 1995

University of California, San Diego
SIO Library 0175C
9500 Gilman Drive
La Jolla, CA 92093-0175

Hancock Library of Biology & Oceanography
Alan Hancock Laboratory
University of Southern California
University Park
Los Angeles, CA 90089-0371

Gifts & Exchanges
Library
Bedford Institute of Oceanography
P.O. Box 1006
Dartmouth, NS, B2Y 4A2, CANADA

Commander
International Ice Patrol
1082 Shennecossett Road
Groton, CT 06340-6095

NOAA/EDIS Miami Library Center
4301 Rickenbacker Causeway
Miami, FL 33149

Research Library
U.S. Army Corps of Engineers
Waterways Experiment Station
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

Institute of Geophysics
University of Hawaii
Library Room 252
2525 Correa Road
Honolulu, HI 96822

Marine Resources Information Center
Building E38-320
MIT
Cambridge, MA 02139

Library
Lamont-Doherty Geological Observatory
Columbia University
Palisades, NY z10964

Library
Serials Department
Oregon State University
Corvallis, OR 97331

Pell Marine Science Library
University of Rhode Island
Narragansett Bay Campus
Narragansett, RI 02882

Working Collection
Texas A&M University
Dept. of Oceanography
College Station, TX 77843

Fisheries-Oceanography Library
151 Oceanography Teaching Bldg.
University of Washington
Seattle, WA 98195

Library
R.S.M.A.S.
University of Miami
4600 Rickenbacker Causeway
Miami, FL 33149

Maury Oceanographic Library
Naval Oceanographic Office
Building 1003 South
1002 Balch Blvd.
Stennis Space Center, MS, 39522-5001

Library
Institute of Ocean Sciences
P.O. Box 6000
Sidney, B.C. V8L 4B2
CANADA

Library
Institute of Oceanographic Sciences
Deacon Laboratory
Wormley, Godalming
Surrey GU8 5UB
UNITED KINGDOM

The Librarian
CSIRO Marine Laboratories
G.P.O. Box 1538
Hobart, Tasmania
AUSTRALIA 7001

Library
Proudman Oceanographic Laboratory
Bidston Observatory
Birkenhead
Merseyside L43 7 RA
UNITED KINGDOM

IFREMER
Centre de Brest
Service Documentation - Publications
BP 70 29280 PLOUZANE
FRANCE

50272-101

REPORT DOCUMENTATION PAGE	1. REPORT NO. WHOI-95-06	2 UOP-95-03	3. Recipient's Accession No.

4. Title and Subtitle
A Processing System for Argos Meteorological Data

5. Report Date
March 1995

6.

7. Author(s) Nancy R. Galbraith

8. Performing Organization Rept. No.
WHOI-95-06

9. Performing Organization Name and Address

Woods Hole Oceanographic Institution
Woods Hole, Massachusetts 02543

10. Project/Task/Work Unit No.

11. Contract(C) or Grant(G) No.
(C) N00014-94-1-0161
(G)

12. Sponsoring Organization Name and Address

Office of Naval Research

13. Type of Report & Period Covered
Technical Report

14.

15. Supplementary Notes
This report should be cited as: Woods Hole Oceanog. Inst. Tech. Rept., WHOI-95-06.

16. Abstract (Limit: 200 words)

Upper Ocean Processes Group meteorological data is transmitted from surface buoys via Argos satellite and processed in an automatic mode on a UNIX workstation. Data is extracted from input files based on instrument type and experiment, processed as appropriate, and plotted, without user intervention. While the processing system normally runs automatically, it is designed so that modules can also be run directly from a terminal when necessary. The Argos processing system allows us to monitor the meteorological data being collected in the field, and provides early information about problems with sensors, instruments, or buoys, when they occur. The automatic process allows more information to be viewed with less effort, and increases the usefulness of the Argos data.

17. Document Analysis a. Descriptors
automatic processing
time-series data
UNIX software

b. Identifiers/Open-Ended Terms

c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Approved for public release; distribution unlimited.	UNCLASSIFIED	60
	20. Security Class (This Page)	22. Price

(See ANSI-Z39.18) *See Instructions on Reverse*

OPTIONAL FORM 272 (4-77)
(Formerly NTIS-35)
Department of Commerce